Exercise 2 – Pekka Lehtola

Task 1. Explain the following terms:

a. Pseudocode

- Pseudokoodi on selkokielellä kirjoitettu koodi tai algoritmi. Pseudokoodin tarkoitus on tehdä koodista helppolukuista ihmiselle, kuitenkin säilyttäen joitain piirteitä kielestä. Pseudokoodi jättää pois kaikki turhat yksityiskohdat jotka ovat välttämättömiä vain ohjelmointi kielelle.

b. Algorithm

- Algoritmi on ohje, kuten esimerkiksi koodin pätkä, jonka tarkoituksena on ratkaista tai laskea joku ongelma. Yksinkertaisuudessansa algoritmi on sarja yksiselitteisiä toimintoja, jonka loppu tuloksena on ratkaistu pulma.

c. Data attribute

- Data-attribuutti on objectin ominaisuus, kuten oheisessa esimerkissä määritetään säde ympyrän attribuutiksi ja alustetaan sen arvoksi nolla.

```
class Circle(object):
    def __init__(self):
        self.radius = 0
```

d. Method

- Metodit ovat objection käytössä olevia funktioita. Kuten tunnilla käydyssä esimerkissä toss oli Coinin metodi.

Task 2. Take a look at the course's assessment (number of accepted exercises meaning certain grade). Write pseudocode for a program where user inputs the number of accepted exercises and program prints out the grade. Use informative and readable output prints.

```
Program start

accepted exercises = user inputs the number of done exercises

if accepted exercises = 13 THEN

        Grade = 5

else if accepted exercises = 12 THEN

        Grade = 4

else if accepted exercises = 11 THEN

        Grade = 3

else if accepted exercises = 10 THEN

        Grade = 2

else if accepted exercises = 9 THEN

        Grade = 1

else

        Grade = 0
```

Task 3. After writing the pseudocode, code task 2. Simple code is enough, no objects needed.

Harjoittelu mielessä koodiin lipsahti objecteja…

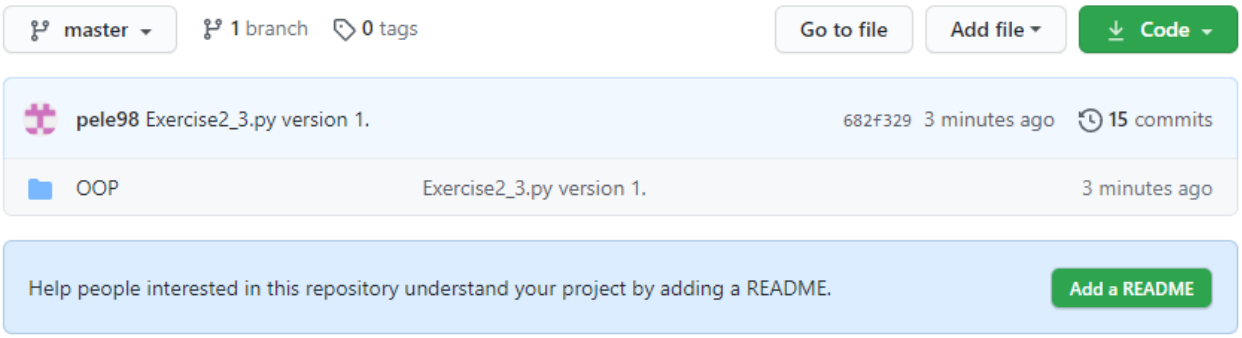
Screen capture of Task 3

```python
#File name: Exercise2_3
#Author: Pekka Lehtola
#Description: Gives grade based on accepted exercises.

class Grading_algorithm:

    # __init__ method intializes data attributes to 0

    def __init__(self):

        self.accepted_exercises = 0
        self.grade = 0


    # num_of_exercises checks if the number of accepted exercises is between 0 and 13
    # if the number is unacceptable the method tells the user whats wrong with the number.
    # method sets number of accepted exercises to given number.

    def num_of_exercises(self, number_of_accepted_exercises):

        if number_of_accepted_exercises < 0:
            self.accepted_exercises = number_of_accepted_exercises
            return print("Number cant be smaller than 0")

        elif number_of_accepted_exercises > 13:
            self.accepted_exercises = number_of_accepted_exercises
            return print("Number cant be greater than 13")

        else:
            self.accepted_exercises = number_of_accepted_exercises


    #If number of accepted exercises is in range 0 to 9 method sets the grade to 0.
    #If the number is in grading scale dictionary method sets grade with the dictionary.
    #if the number of accepted exercises is anything else, the grade is set as "Undefined".

    def grading(self):

        grading_scale = {13:5, 12:4, 11:3, 10:2, 9:1}

        if self.accepted_exercises in grading_scale:

            self.grade = grading_scale[self.accepted_exercises]
            return print("Your grade is", self.grade, end="\n\n")

        elif self.accepted_exercises in range(0, 9):
            self.grade = 0
            return print("Your grade is", self.grade, end="\n\n")

        else:
            self.grade = "Undefined"
            return print("Your grade is", self.grade, end="\n\n")


#Main function that takes the number of accepted exercises from user.

def main():

    my_grade = Grading_algorithm()

    my_grade.num_of_exercises(int(input("Input the number of accepted exercises: ")))

    my_grade.grading()

while True:

    main()
```

Screen capture of the output of Task 3

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe
Input the number of accepted exercises: -1
Number cant be smaller than 0
Your grade is Undefined

Input the number of accepted exercises: 14
Number cant be greater than 13
Your grade is Undefined

Input the number of accepted exercises: 6
Your grade is 0

Input the number of accepted exercises: 12
Your grade is 4

Input the number of accepted exercises:
```

Screen capture of the Git status after Task 3



Task 4. Write pseudocode for a program that accepts student's name and grade as input and counts the average of grades of all students. If you have difficulties, you can fix the number of students to e.g. 5. Print out the average. Use informative and readable output prints

Program start

create empty list of all students

loop until stopped:

students name = input from user

students grade = input from user

combine students name with given grade

add student to list of all students

count the sum of all grades in the list

divide the sum with number of students.

return grade average

Task 5. After writing the pseudocode, code task 4. Simple code is enough, no objects needed.

Screen capture of Task 5

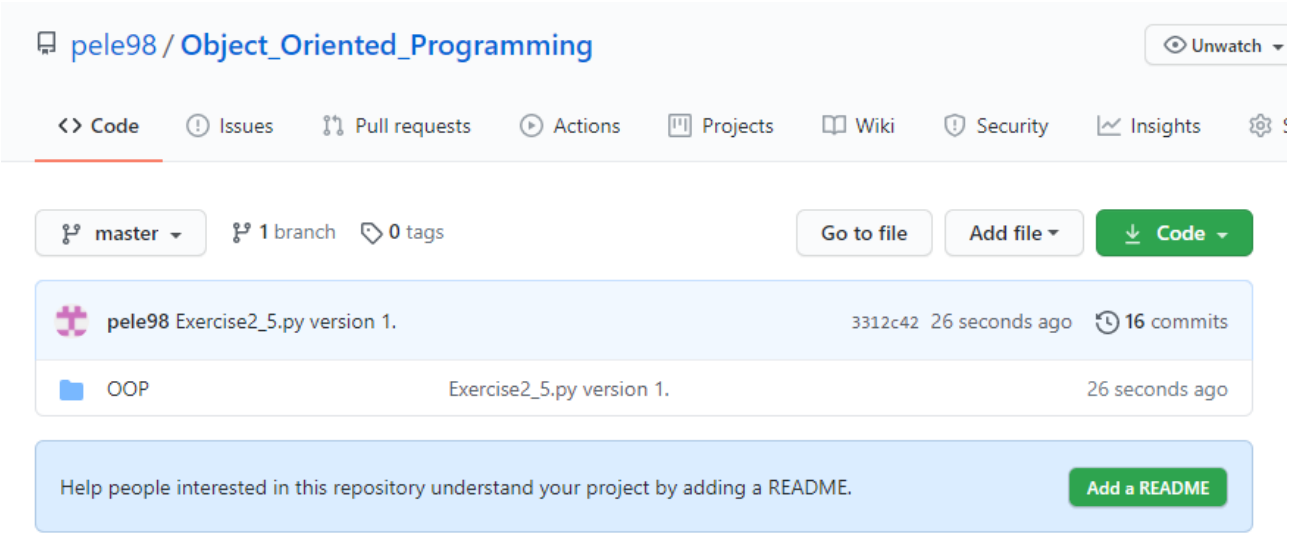
```python
#File name: Exercise2_5
#Author: Pekka Lehtola
#Description: Counts grade average from all students. New students can be added.


class Student:

    #Initializing student with attributes

    def __init__(self, name, grade):

        self.name = name
        self.grade = grade

class Class:

    #Setting up class named Class.

    def __init__(self, class_name):

        self.class_name = class_name
        self.sum_of_grades = 0
        self.class_average = 0
        self.students_in_class = []


    # Appends the list of students in class wit students name.
    # Calculates new grade sum with new students grade.
    # Calculates class average with new sum.

    def add_student_to_class(self, student):

        self.students_in_class.append(student.name)
        self.sum_of_grades += student.grade
        self.class_average = self.sum_of_grades / len(self.students_in_class)

    def print_average(self):

        return print(self.class_average)
```

```python
def main():

    # Convert user input to Class object with user given name.

    user_input_for_class_name = str(input("Give name to the class: "))
    class_room = Class(user_input_for_class_name)

    while True:

        #Uses user input to navigate options.
        user_input = str(input("add students, class average or exit: "))

        if user_input == "add students":

            while True:

                new_students_name = input("Give name to the student or exit: ")

                #If the name is "exit" returns back to main function.
                if new_students_name == "exit":
                    break

                new_students_grade = int(input("Give a grade to the student: "))

                # Create new student object with user given name and grade.
                new_student = Student(new_students_name, new_students_grade)

                #Adds the new student to Class, calculate sum and class average
                class_room.add_student_to_class(new_student)

                #Printing confirmation of succesful addition and prints out current students in class room.
                print(new_student.name, "added to classroom ", class_room.class_name)
                print("Classroom consists from the following students", class_room.students_in_class)

        #Printing class average.
        elif user_input == "class average":

            class_room.print_average()

        #Ends program
        elif user_input == "exit":

            exit()

        else:

            continue

main()
```

Screen capture of the output of Task 5

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive
Give name to the class: Ptivis19o
add students, class average or exit: add students
Give name to the student or exit: Pekka
Give a grade to the student: 5
Pekka added to classroom  Ptivis19o
Classroom consists from the following students ['Pekka']
Give name to the student or exit: Jaakko
Give a grade to the student: 2
Jaakko added to classroom  Ptivis19o
Classroom consists from the following students ['Pekka', 'Jaakko']
Give name to the student or exit: Linda
Give a grade to the student: 4
Linda added to classroom  Ptivis19o
Classroom consists from the following students ['Pekka', 'Jaakko', 'Linda']
Give name to the student or exit: Tomi
Give a grade to the student: 4
Tomi added to classroom  Ptivis19o
Classroom consists from the following students ['Pekka', 'Jaakko', 'Linda', 'Tomi']
Give name to the student or exit: exit
add students, class average or exit: class average
3.75
add students, class average or exit: exit

Process finished with exit code 0
```

Screen capture of the Git status after Task 5



Task 6. Imagine you would have to code a simple alarm clock (shows time and alarms you at certain time you can set). Which data attributes will you have? Do the attributes have some value restrictions? You should find at least 5 data attributes. Which methods would you need? Which methods should be public and which ones should be private?

Luokan atribuutit: Kellolle tunnit, minuutit, sekuntit . Herätys kellolle: Onko herätys aktiivinen, hälyttääkö se ja herätyksen aika. Metodeja olisi ainakin herätyksen ajan asettaminen, herätyksen päälle ja pois laittaminen. Käyttäisin myös Pythonin Time metodia. Julkisiksi metodeiksi jättäisin herätysten asettamiset. Yksityisiksi jättäisin ajan saamisen.

Task 7. Take a look at the coin.py, write it down in your IDE and run it. See that coin gets tossed.

Screen capture of Task 7

```python
#File: Example
#Description: Simulate coin flip
#Author: Pekka Lehtola

import random

class Coin:
    #The __init__ method initializes the sideup data attribute with "heads"

    def __init__(self):
        self.sideup ="Heads"

    # The toss method generate a random number
    # int the range of 0 trough 1, If the number 0, the sideup is set to "Heads"
    #otherwise, sideup is set to "Tails"

    def toss(self):
        if random.randint(0, 1) == 0:
            self.sideup = "Heads"
        else:
            self.sideup = "Tails"

    # The get_sideup method return the value referenced by sideup.

    def get_sideup(self):
        return self.sideup

# The main function

def main():

    #Create an object from the Coin class.
    my_coin = Coin()

    # Display the side of the coin that is facing up.
    print("This side is up: ", my_coin.get_sideup())

    #Toss the coin.
    print("I am tossing the coin...")
    my_coin.toss()

    # Display the side of the coin that is facing up.
    print("This side is up: ", my_coin.get_sideup())


# Call the main function.
main()
```

Screen capture of the output of Task 7

```
C:\Users\pekka\AppData\Local\Microsoft\Win
This side is up:  Heads
I am tossing the coin...
This side is up:  Tails

Process finished with exit code 0
```

Screen capture of the Git status after Task 7

master / Object_Oriented_Programming / OOP / Excercise2 / Example.py / Jump to

pele98 Excample code made by Anne.

1 contributor

47 lines (32 sloc) | 1.06 KB

```
#File: Example
#Description: Simulate coin flip
#Author: Pekka Lehtola

import random

```

Task 8. Modify the toss_the_coin() function so that there are 2 more options: Coin lands on the table upright (and not flat showing heads or tails) or coin drops on the ground and disappears (on a rabbit hole). Name the options properly and give informative and readable output of the status.

Screen capture of Task 8

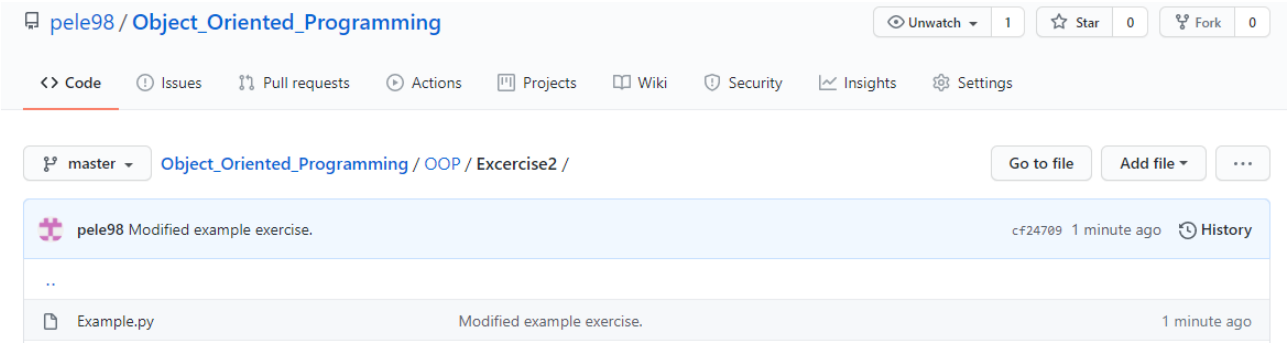
```
#File: Example
#Description: Simulate coin flip
#Author: Pekka Lehtola

import random

class Coin:
    #The __init__ method initializes the sideup data attribute with "heads"

    def __init__(self):
        self.sideup ="Heads"


    #Selects a random integer between 0 and 14. If the value is 0-4 side up is heads.
    #If the value is 5-9 tails is choosen. 10-12 coin lands upright.
    #13-14 coin falls into an rabbit hole...

    def toss(self):

        random_number = random.randint(0,14)

        if random_number in range(0,5):
            self.sideup = "Heads"

        elif random_number in range(5,10):
            self.sideup = "Tails"

        elif random_number in range(10, 13):
            self.sideup = "Coin landed upright"

        else:
            print("Coin landed into a rabbit hole...Game over, because you don't have a coin anymore.")
            exit()

    # The get_sideup method return the value referenced by sideup.

    def get_sideup(self):

        return self.sideup

# The main function

def main():

    #Create an object from the Coin class.
    my_coin = Coin()
```

```python

    # Display the side of the coin that is facing up.
    print("This side is up: ", my_coin.get_sideup())

    print("To toss the coin press enter.", end="\n\n")

    while True:

        #Makes the user press enter before next toss.
        user_input = input()

        #Toss the coin.
        print("I am tossing the coin...")
        my_coin.toss()

        # Display the side of the coin that is facing up.
        print("This side is up: ", my_coin.get_sideup())


# Call the main function.
main()
```

Screen capture of the output of Task 8

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive
This side is up:  Heads
To toss the coin press enter.


I am tossing the coin...
This side is up:  tails

I am tossing the coin...
This side is up:  Heads

I am tossing the coin...
This side is up:  Coin landed upright

I am tossing the coin...
This side is up:  tails

I am tossing the coin...
This side is up:  Heads

I am tossing the coin...
This side is up:  Heads

I am tossing the coin...
Coin landed into a rabbit hole...Game over, because you don't have a coin anymore.

Process finished with exit code 0
```

Screen capture of the Git status after Task 8



Task 9. Write pseudocode for the alarm clock (see task 6.)

Program starts

Initialize Clock:

Hours = get current hours

Minutes = get current minutes

Seconds = get current seconds

Alarm_time = 8:00

Alarm_set = False

Alarm_sound = False

Defining update_clock:

Hours = get current hours

Minutes = get current minutes

Seconds = get current minutes

Defining setting_alarm:

Alarm_time = user input

Alarm_set = True

Defining turning_alarm_off:

Alarm_set = False

Alarm_sound = False

```
Main loop:
        Wait 1 second
        update_clock
        If keyboard pressed(space):
                If Alarm_set = False:
                        setting_alarm
                Else:
                        turning_alarm_off

        else if alarm_set = True:
                If alarm_time = current time:
                        alarm_sound = True
        else if alarm_sound = True:
                print out “Its time to wake up!”

Start main loop
```

Task 10. Code the alarm clock, use objects.

Screen capture of Task 10

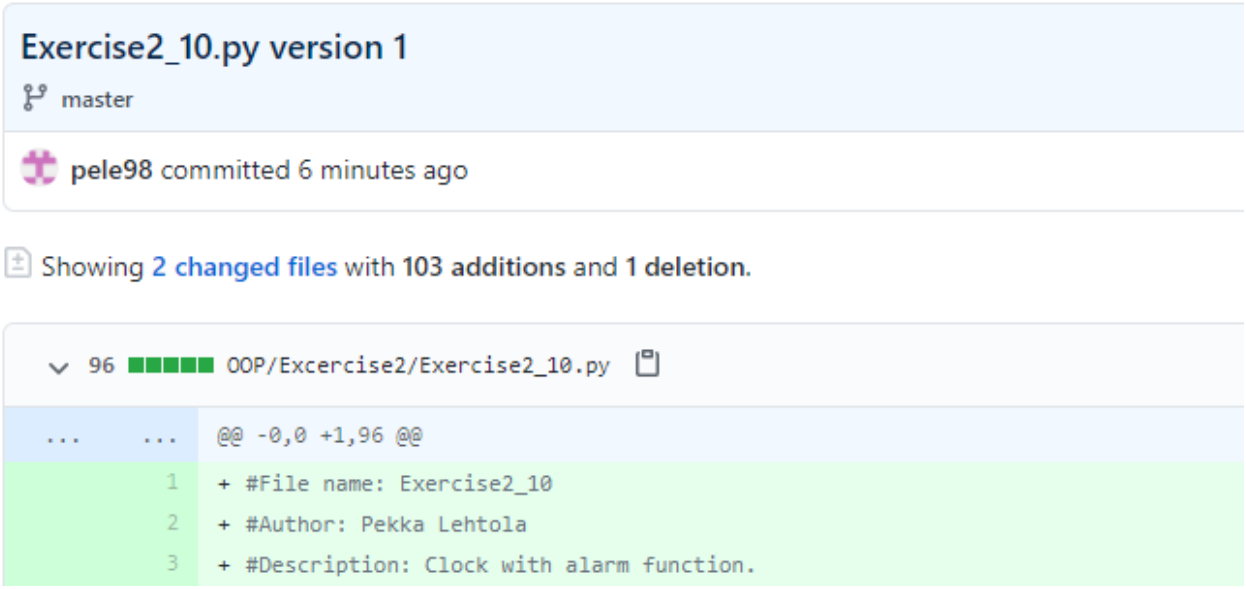
```python
#File name: Exercise2_10
#Author: Pekka Lehtola
#Description: Clock with alarm function.


#Importing time for local tima, keyboard for setting up alarm and playsound
#To playback alarm sound
import time
import keyboard
from playsound import playsound

class Clock:

    def __init__(self):

        #Hour, minute and seconds set as local time. Alarm turned intialy
        #off with alarm time set to 00:00.
        self.hour = int(time.strftime("%H"))
        self.minute = int(time.strftime("%M"))
        self.second = int(time.strftime("%S"))
        self.alarm_time_hour = 0
        self.alarm_time_minute = 0
        self.alarm_set = False
        self.alarm_sound = False

    #Used for updating clocks time.
    def update_clock(self):

        self.hour = int(time.strftime("%H"))
        self.minute = int(time.strftime("%M"))
        self.second = int(time.strftime("%S"))

    #Takes time input from user and sets it as alarms time.
    #Prints out confirmation that alarm was set to users input.
    #Try used for avoiding crash if ValueError occours
    def setting_alarm(self):

        try:
            self.alarm_time_hour = int(input("Set up alarms hour: "))
            self.alarm_time_minute = int(input("Set up alarms minute: "))
            self.alarm_set = True
            print("Alarm set to: ", str(self.alarm_time_hour) + ":" + str(self.alarm_time_minute))

        except:
            print("Value error", end="\n\n")
            print("Time is: ", str(clock.hour) + ":" + str(clock.minute))

    #Sets alarm back to initial values.
    def turning_alarm_off(self):

        self.__init__()
        print("Alarm turned off at", str(clock.hour) + ":" + str(clock.minute))

clock = Clock()
```

```python
#Printing out starting time and instruction for setting alarm.
print("To setup alarm press SPACE. Hold down SPACE to end alarm.", end="\n\n")
print("Time is: ", str(clock.hour) + ":" + str(clock.minute))


#Main function
def main():

    while True:

        #Detects if space is pressed. Depending if alarm is on or not
        #sets up alarm status.
        if keyboard.is_pressed("space"):

            if not clock.alarm_set:

                clock.setting_alarm()

            else:
                clock.turning_alarm_off()
                time.sleep(1)

        # If alarm is triggered plays alarm sound and prints out cheerful
        # message.Time.sleep used for avoiding alarm getting annoying.
        if clock.alarm_sound:
            print("Its time to wake up!")
            playsound("alarm.mp3")
            time.sleep(0.4)

        #Checks if its alarm time.
        if int(clock.alarm_time_hour) == (clock.hour):
            if int(clock.alarm_time_minute) == clock.minute:
                clock.alarm_sound = True

        #Checks if local time has changed, if yes updates the clock.
        #Prints out clock every minute.
        if clock.minute != int(time.strftime("%M")):
            clock.update_clock()
            print("Time is: ", str(clock.hour) + ":" + str(clock.minute))

main()
```

Screen capture of the output of Task 10

```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe
To setup alarm press SPACE. Hold down SPACE to end alarm.

Time is:  12:56
Set up alarms hour: 12
Set up alarms minute: 59
Alarm set to:  12:59
Time is:  12:57
Time is:  12:58
Time is:  12:59
Its time to wake up!
Its time to wake up!
Its time to wake up!
Its time to wake up!
Its time to wake up!
Alarm turned off at 12:59
```

Screen capture of the Git status after Task 10



Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Harjoitukset olivat hiukan haastavia, mutta vain siksi kun tein myös tehtävät 5 ja 3 käyttäen olioita.

Doing this exercise, I learned…

Olioiden käytöstä koodissa ja lisäksi oppisin käyttämään TRY funktiota välttääkseni esimerkiksi Value errorit. Lisäksi keksin kätevän tavan resetöidä olion ajamalla __init__ komennon uudestaan.

I am still wondering…

Yhdessä vaiheessa koodia jouduin käyttämään __str__ ja __repr__ metodeita, mutta käyttötarkoitus jäi vähän epäselkeäksi ja onnistuin välttämään niiden käytön. Oletan kuitenkin että tulevat vastaan vielä jossain kohtaa kurssia tai koodaus uraa.

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

Alkuperäinen suunnitelma oli tehdä herätyskello käyttäen tkinter kirjastoa, mutta loppui ikävä kyllä aika.

Pseudo koodin kirjoitus oli mielestäni haastavaa, sillä löysin netistä satoja erilaisia tapoja kirjoittaa sen. Lopputuloksena omasta mielestä minun Pseudokoodini oli aika epäselkeä.